CSE 3105/ CSE 3137

OBJECT ORIENTED ANALYSIS AND DESIGN

FALL 2020

COURSE PROJECT: "*Media Browser Application*"

System Design Document

Group 10

Burak Abdulbaki ULU – 170315017

Halil Yusuf KAPLAN – 170315064

Muhammed Serhat BOZKURT –170315039

Semih DÖNMEZ – 170315069

Yunus Emre KARADAŞ – 170315054

17 January 2021

Table of Contents

1 Introduction

Considering the class diagram and sequence diagrams that form the basis of our system, which we mentioned in our previous report (RAD V3[1]) and UML diagrams[2]; at the same time, we decided to implement the Model-View-Controller (MVC) software architecture, as it largely complies with our design goals.

1.1 Purpose of the System

Our aim is to enable users to easily organize and use the media they want, with a practical and understandable interface; It is to provide an application experience that includes functions such as the user to play a media easily and to add / remove it to playlists, to delete it.

1.2 Design goals

Ease of Use: The system's features should be easy [generalization of nonfunctional requirement 1].

User-friendliness: All users can easily use the application [generalization of nonfunctional requirement 2].

Well-defined Interfaces: The user interface should be simple and understandable [generalization of nonfunctional requirement 3].

Robustness: In case of failure, the system should be restarted without any data loss [generalization of nonfunctional requirement 4].

Ease of Learning: First-time tutorial should be provided to the user [generalization of nonfunctional requirement 4].

High-performance: Logging in to system should not exceed 5 seconds [generalization of nonfunctional requirement 6].

Modifiability: New features should be able to be added without affecting other features in the software [generalization of nonfunctional requirement 9].

Security: Data belonging to different user types cannot be accessed by other users [deduced from application domain].

Reusability: Parts to be made must be reused [derived from the logic of the object oriented programming].

Good documentation: Before implementing the system, the system should be fully discussed, designed and documented.

Readability: In the whole process; reports, coding standards, etc. should be easily understood and analyzed later by those who review.

TRADE-OFFS

Modifiability vs. Reusability

As new features are added to a reusable part, people who previously used that part have to add the updated part every time it is updated.

PRIORITY

High Priority:

- Readability
- Good Documentation
- Ease of Use
- Robustness
- Modifiability

Medium Priority:

- Security
- User-friendliness
- Reusability
- Well-defined Interfaces

Low Priority:

- Ease of Learning
- High-performance

2 Current Software Architecture

VLC – Modular Architecture[3]

Windows Media Player – Media Foundation(Model – View - Controller) [4]

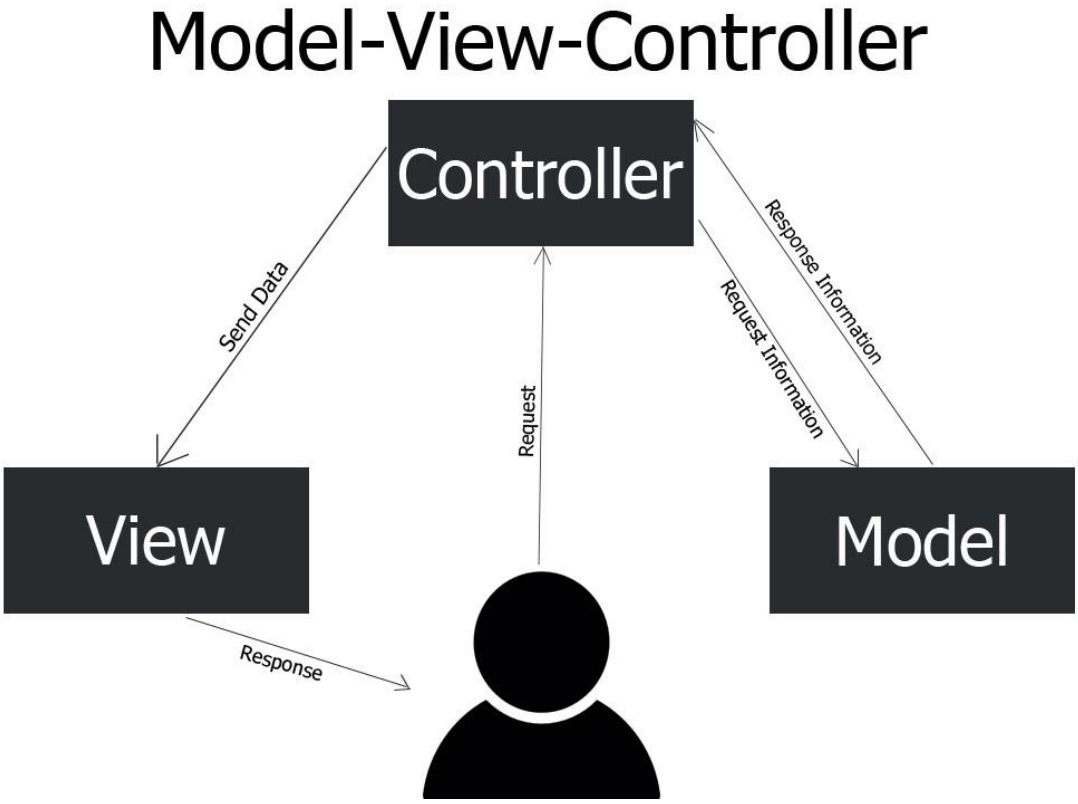
- Lowering the “Coupling”.

Films & TV – Media Foundation(Model – View - Controller) [4]

- Lowering the “Coupling”.

Since we couldn't find the GOM Player and Celluloid architectures, we couldn't add them here. We think these applications have been translated.

The second section, *Current software architecture*, describes the architecture of the system being replaced. If there is no previous system, this section can be replaced by a survey of current architectures for similar systems. The purpose of this section is to make explicit the background information that system architects used, their assumptions, and common issues the new system will address.

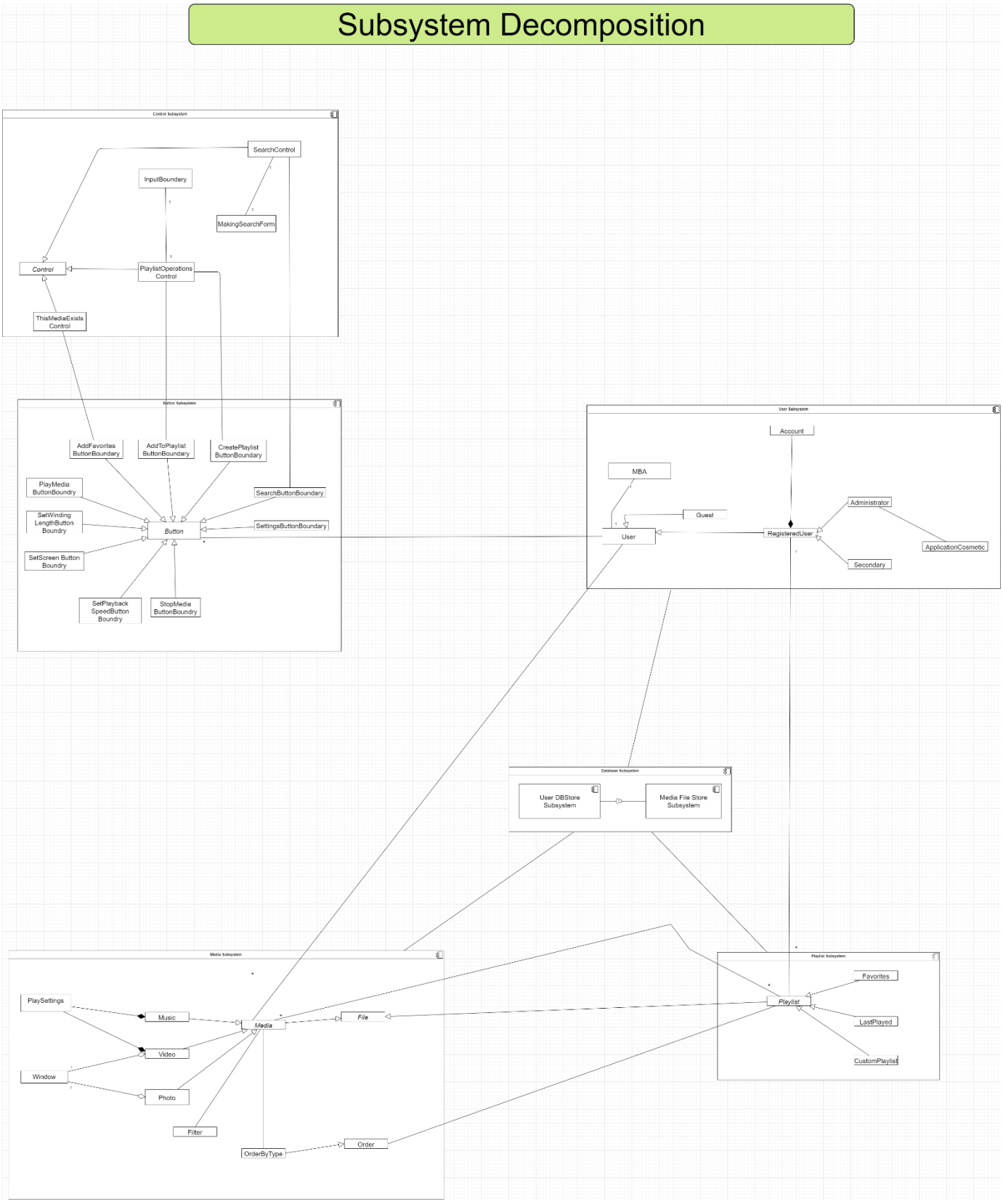


Representation of Model – View – Controller(MVC) Architecture

3 Proposed Software Architecture

The third section, *Proposed Software Architecture*, documents the system design model of the new system. It is divided into five subsections:

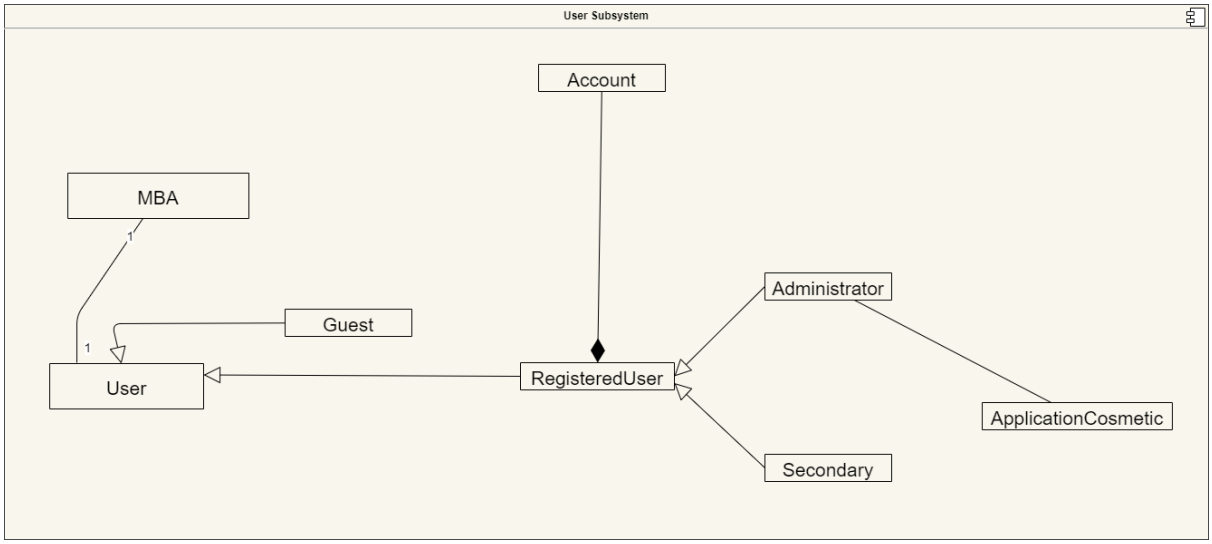
3.1 Subsystem decomposition



SUBSYSTEMS :

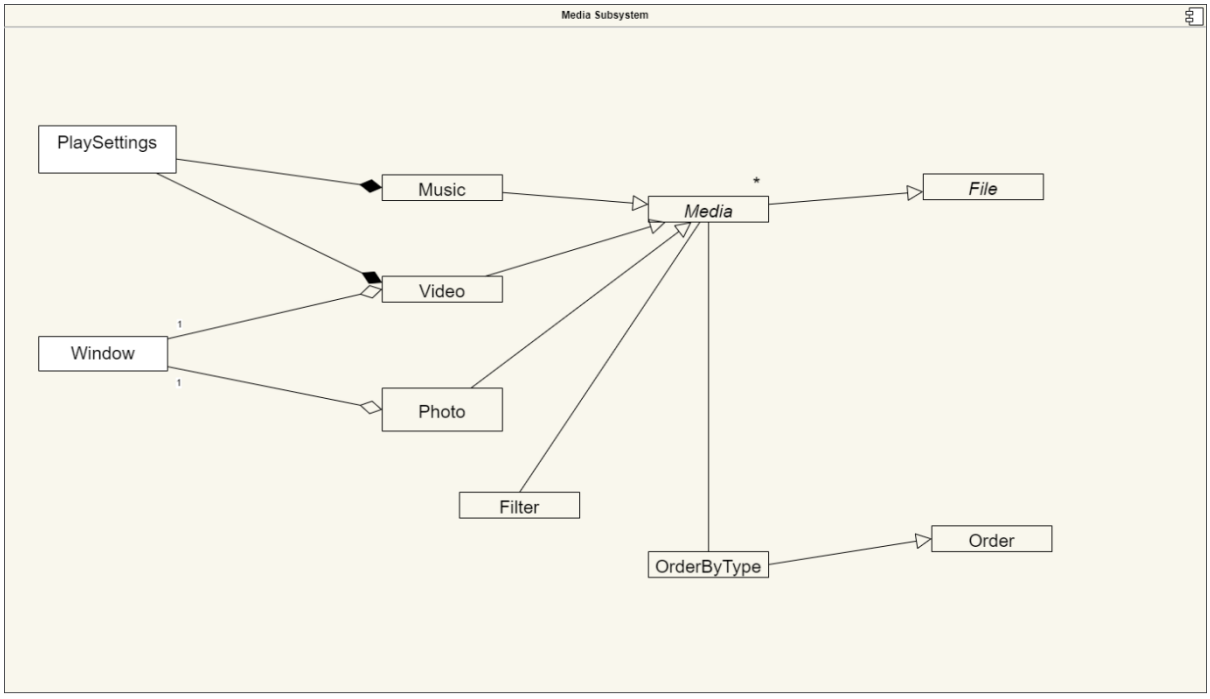
User Subsytem :

"User Subsystem" is responsible for the login and use of the application. "User Subsystem" can use the "Playlist Subystem" via registered user.



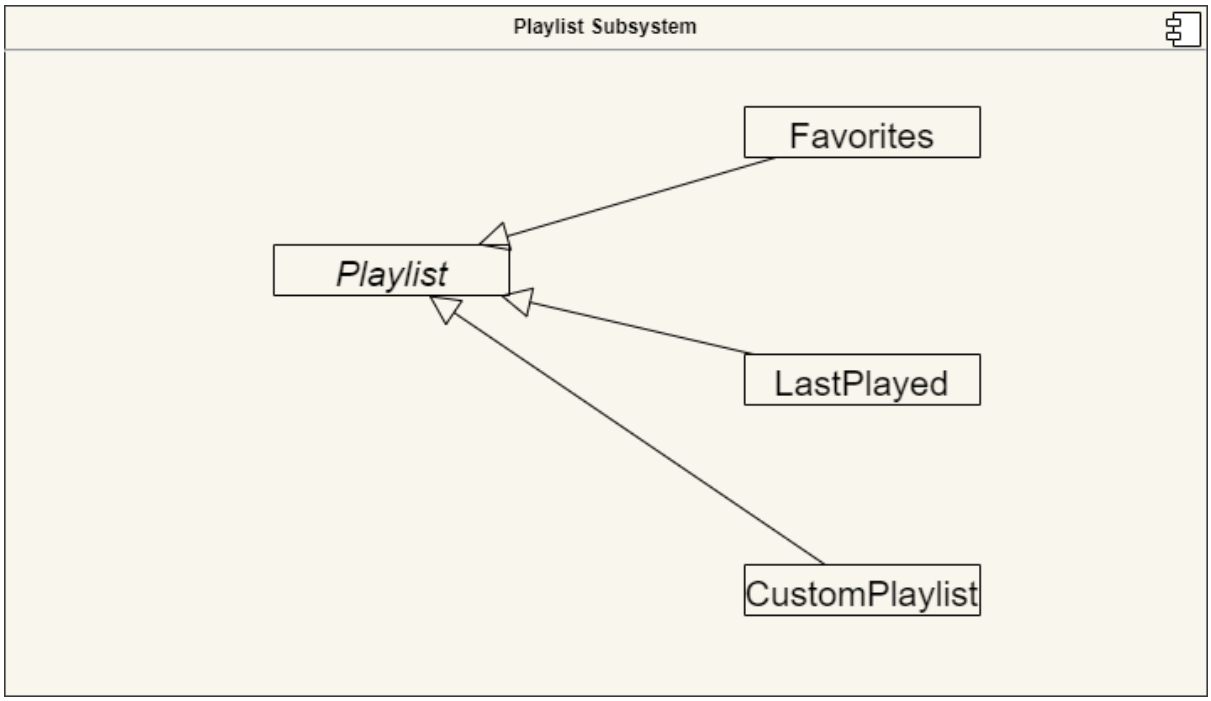
Media Subsytem :

"Media Subsystem" is responsible for the execution and different operations of the media in the application. Adding and deleting a new media can be done by Administrator User.



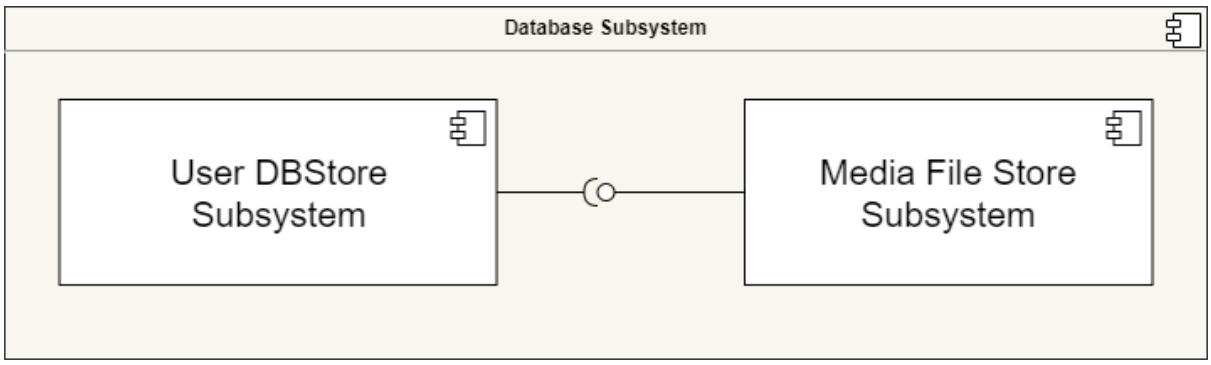
Playlist Subsytem :

It is responsible for the operations such as adding - deleting selected media to the playlist, creating a new playlist from scratch, or deleting and changing the created playlists. In addition, it is also responsible for adding and removing the desired media to the favorite list.



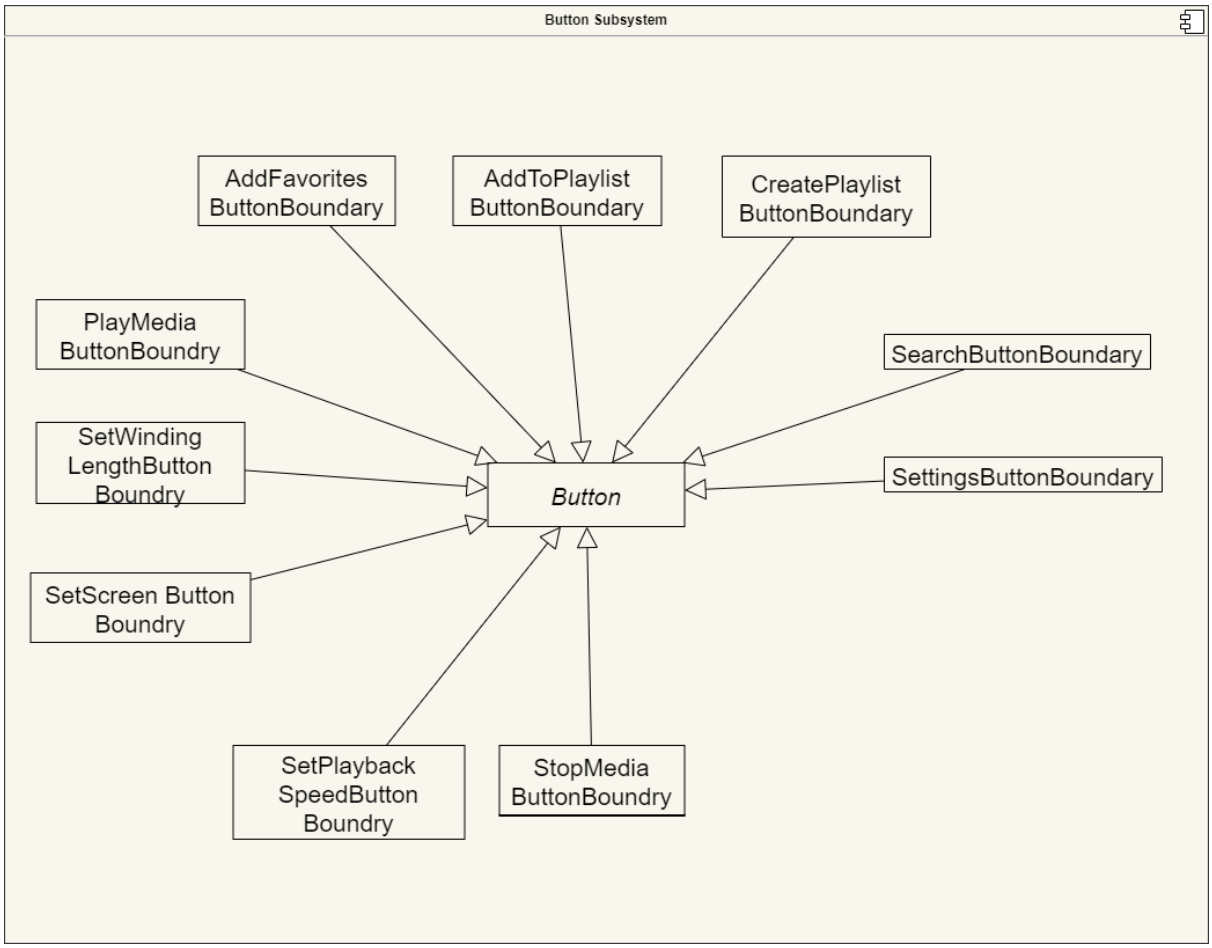
Database Subsytem :

Database subsystem stores all of the photos, videos and music in the application. It carries out the processes of adding, deleting and changing on these media. It also stores user information.



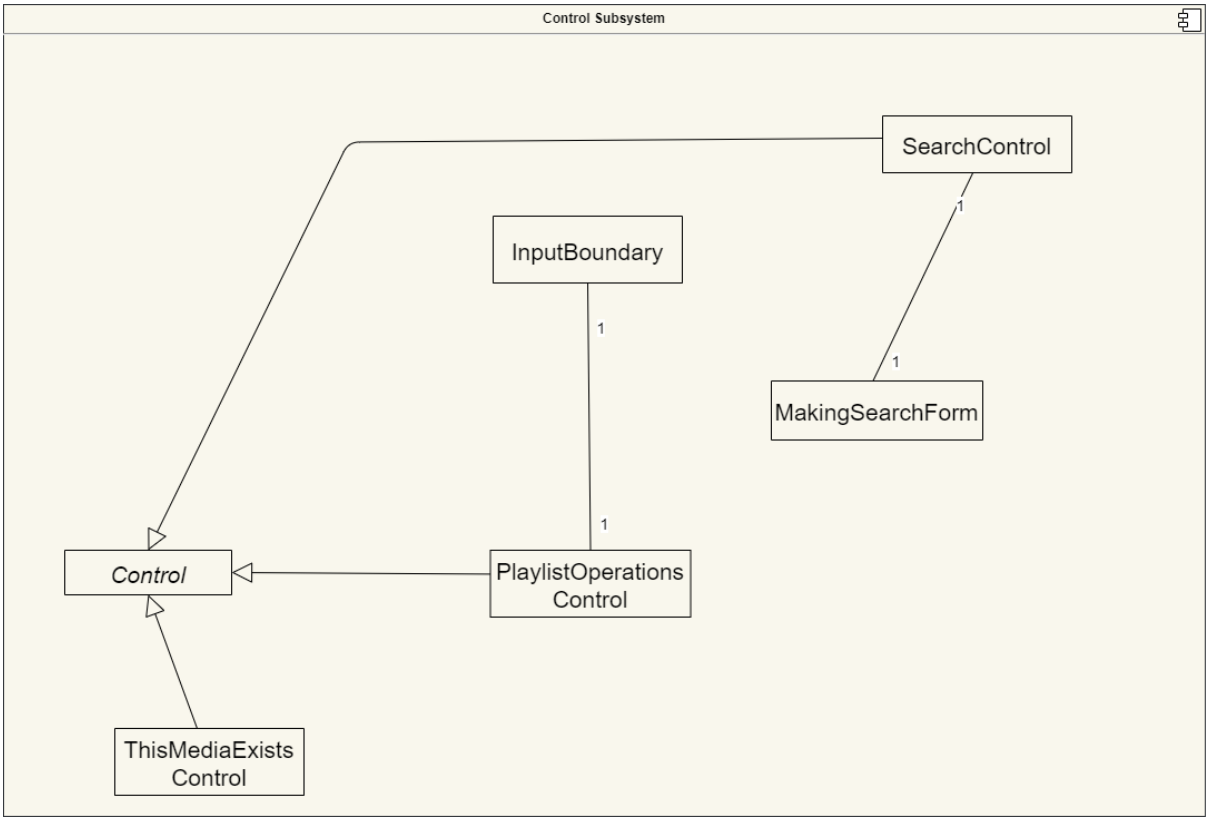
Button Subsytem :

It is responsible for the user's communication with the Application. Notifies the user's requests to the control object.



Control Subsytem :

It is responsible for checking and fulfilling the Button Subsystem requests.



ARCHITECTURE :

Design Goals for Decide the Architecture :

- *User-friendliness*
- *Well-defined Interfaces*
- *Modifiability*
- *Good documentation*
- *Readability*

What is our system like ?

In this application, entity, boundary and control objects; In line with the purpose of the system, they interact with each other very often and as a result of this interaction, they communicate with each other. Especially in line with our "User-friendliness" and "Well-defined Interfaces" design purposes, it is among our main objectives to express these processes explicitly and clearly to the user. Which architecture should be chosen in line with these?

Why these architectures not fit for our system ?

- "Layered Architecture" and "Repository Architecture" :

  Architectures that are more suitable for application designs such as layered architecture or repository architecture, whose purpose is to fulfill specific functions or to serve other systems connected to it, were not preferred.

- "Client/Server" and "Peer – to – peer"

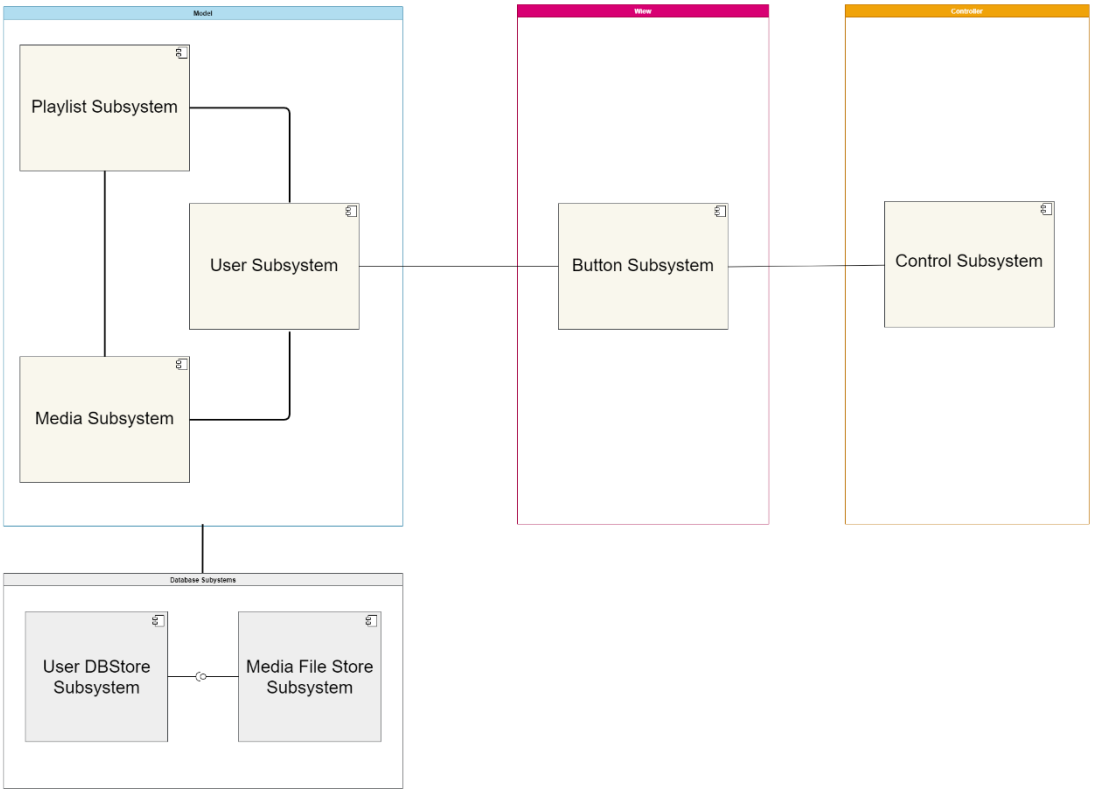
  Since we do not interact with any internet or network in this application, architectures in which processes such as client - server, peer - to - peer architecture are carried out mostly over network systems are also out of our options.

Which architecture will be using in our system ?

In addition, we plan to create a system that can be easily understood by developers and analysts who want to examine and analyze the system in line with our "Good Documentation" and "Readability" design goals.

As a result, in our application where the user interface (boundary) objects are at the forefront, the relation between the objects is used intensely, and obeys the object - oriented logic; It was decided to use ***"Model - View - Controller (MVC)"*** architecture, which provides a clearer and more specific classification.

With "Model - View - Controller" Architecture



3.2 Hardware/software mapping

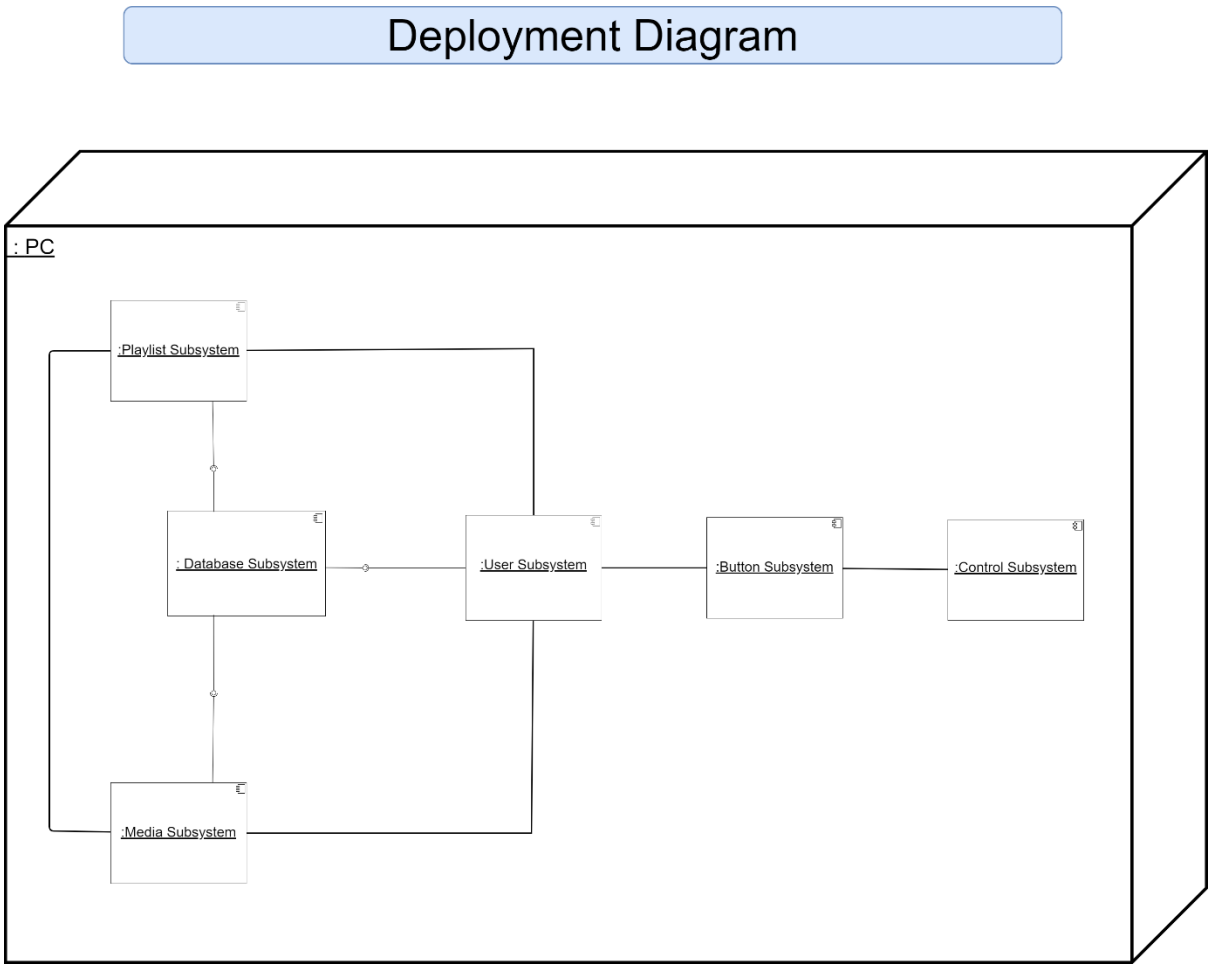
The application is designed to work on the local pc, so it does not need to connect to the internet. A single pc (personal computer) and a database will be sufficient for the application.

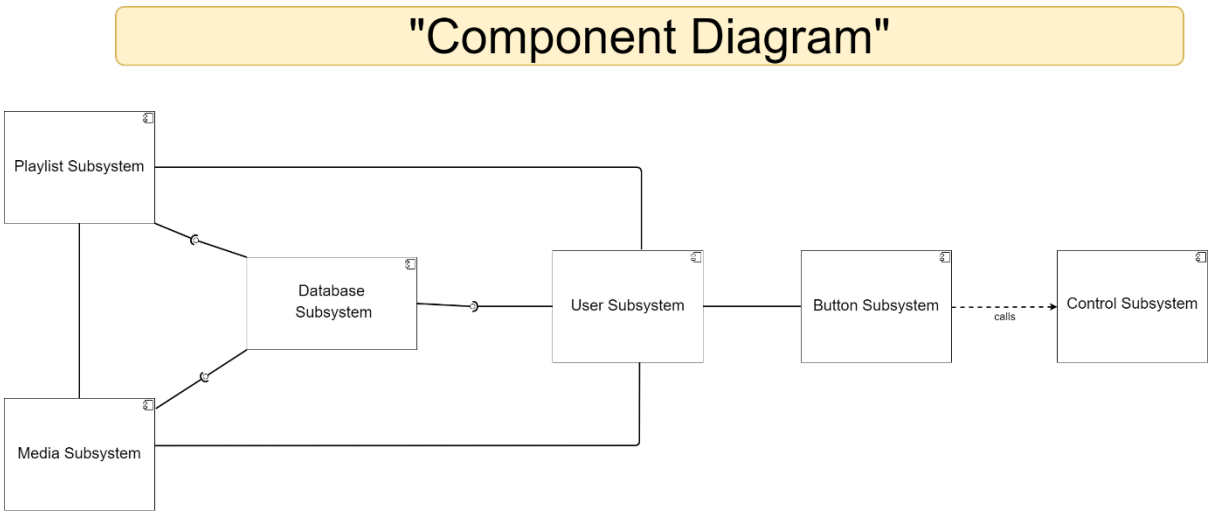
How will we implement subsystems: With hardware or with software?

We need both in our subsystems.

"User Subsytem", "Playlist Subsytem", "Media Subsystem" and "Database Subsystem" are in the area of my hardware especially since they store data in memory and therefore deployment is shown in the diagram.

In addition to these, "Button Subsytem" and "Control Subsystem" are components of the software part, since they carry out all software operations and only operate when the application is open.





How do we map the object model onto the chosen hardware and / or software?

- ❖ Memory:

"Playlist Subsytem", "Media Subsystem" and "Database Subsystem" use memory to store data.

- ❖ Input / Output:

Input operations are done with "Button Subsytem", output operations are done with "Control Subsytem".

3.3 Persistent data management

Permanent data in the application we designed are:

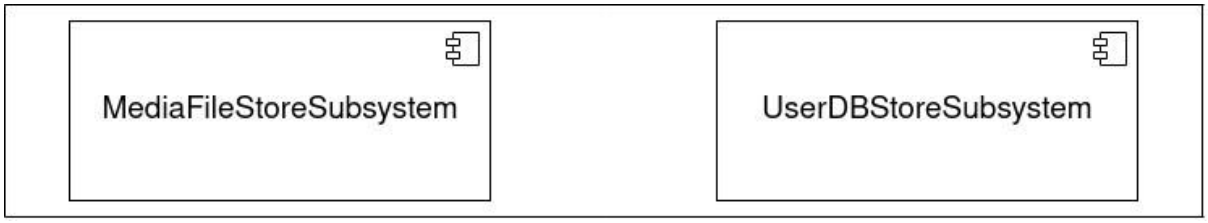
- Media
- Account
- Favorites
- LastPlayed
- CustomPlaylist
- PlaySettings
- ApplicationCosmetic

Permanent data management system that we determined is a hybrid system. In other words, it is a system that includes both file system and relational database. The reasons for choosing such a system are as follows:

The file system is more advantageous in that medias can be added to the system more quickly. Because if we use only database, adding media to the system will be tiring in terms of the application. Also, database queries will run slower due to store all media in the database will increase database size. In addition, store medias in file system will be more advantageous due to reading speed of the discs is faster than reading speed of the database. On the other hand, store users informations in the relational database system provide quick access to the datas. Also, there is no data repetition because datas is stored with relational. Hence, using hybrid system is more advantageous in terms of storage. In brief, using hybrid system is more advantageous from the point of especially performance.

There are medias that added by administrator user in the file system. This file system must not be able to be influenced from the outside. In other words, media can be added by administrator user to the system from only single interface (from within application).

There are user account informations, informations of medias that added by user to the favorites playlist, informations of media that last played by user, informations of playlist that created by user and informations of settings that determined by user in the playing media interface in the relational database system. Also, this system includes informations about application interface cosmetic that determined by administrator user. The subsystems we have identified are as follows:



MediaFileStoreSubsystem: All data that the administrator user adds to the system is stored in this subsystem. Other users can access the medias in the system with use this subsystem.

UserDBStoreSubsystem: This subsystem is responsible for storing data belong to all user. The users communicates with the subsystem to do any operation in the application

3.4 Access control and security

Users that except Guest User login the system using their username and password. The Guest User logins into the system directly. Authentication with username and password takes place in the local database. The user writes the user name and password in the relevant places to enter the system. The system communicates with the local database to authentication process. If an account belonging to the username entered by the user is defined, it is verified with the password entered in the relevant place. If all information matches, login to the system is successful.

Users added the system by administrator user. Added user login into the system with information that given by administrator user. If user wants to change own password, he/she can perform changing password process from relevant section in the system. When the user changes own password, the system encrypts the password specified by the user with the SHA-256 encryption algorithm and stores it in the local database.

Since the login information of the user will be kept on the same hardware, we found it appropriate to encrypt the user passwords with the SHA-256 encryption algorithm. Doing this in this way brings us closer to our "security" goal which is one of our design goals.

Access Matrix :

	Playlist	**Favorites**	**CustomPlaylist**	**File**
Administrator User	createNewPlaylist() deletePlaylist() addMediaToPlaylist() deleteMediaFromPlaylist()	addMediaToFavorites() deleteMediaFromFavorites()	<<create>> changePlaylistName()	search() select()
Secondary User	createNewPlaylist() deletePlaylist() addMediaToPlaylist() deleteMediaFromPlaylist()	addMediaToFavorites() deleteMediaFromFavorites()	<<create>> changePlaylistName()	search() select()
Guest User				search() select()

	Order	**OrderByType**	**Filter**	**Media**
Administrator User	aZ() zA() date() favorite()	type()	onlyMusic() onlyVideo() onlyPhoto() all()	<<create>> addMedia() deleteMedia()
Secondary User	aZ() zA() date() favorite()	type()	onlyMusic() onlyVideo() onlyPhoto() all()	
Guest User	aZ() zA() date()	type()	onlyMusic() onlyVideo() onlyPhoto() all()	

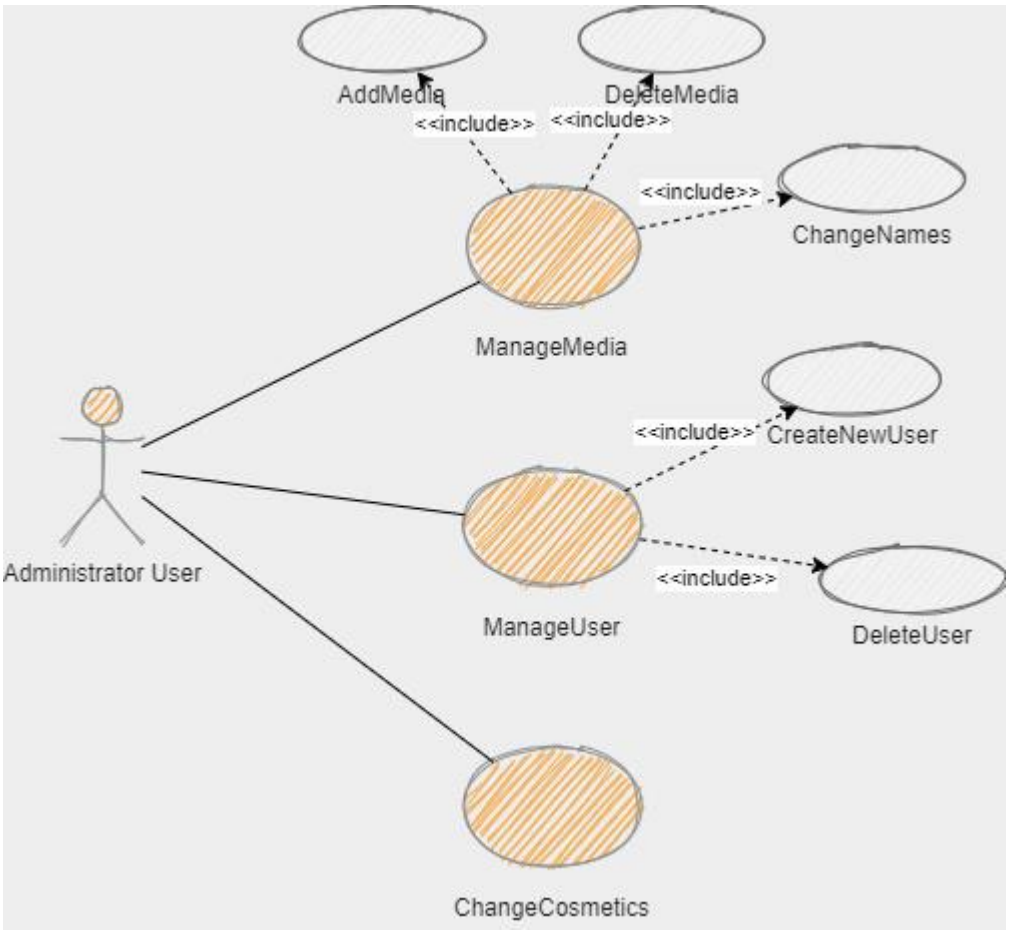
	PlaySettings	Window	PlaylistOperationsControl	MakingSearchForm
Administrator User	setPlaybackSpeed() setWindingLength() playMedia() stopMedia()	setScreenSize() setZoomIn-Out()	choosePlaylist()	setMediaType(type) enterMediaName(chrctrList)
Secondary User	setPlaybackSpeed() setWindingLength() playMedia() stopMedia()	setScreenSize() setZoomIn-Out()	choosePlaylist()	setMediaType(type) enterMediaName(chrctrList)
Guest User	setPlaybackSpeed() setWindingLength() playMedia() stopMedia()	setScreenSize() setZoomIn-Out()		setMediaType(type) enterMediaName(chrctrList)

	Account	ApplicationCosmetic	PlayMediaButtonBoundary	StopMediaButtonBoundary
Administrator User	loginAsAdministrator() logOut()	changeCosmetic()	press()	press()
Secondary User	loginAsSecondaryUser() logOut()		press()	press()
Guest User	continueAsGuest()		press()	press()

	SetWindingLength ButtonBoundary	SetScreenButtonBoundary	SetPlaybackSpeed ButtonBoundary	SearchButtonBoundary
Administrator User	press()	press()	press()	press()
Secondary User	press()	press()	press()	press()
Guest User	press()	press()	press()	press()

	AddFavorites ButtonBoundary	AddToPlaylist ButtonBoundary	CreatePlaylist ButtonBoundary	MBA
Administrator User	press()	press()	press()	search(type, name)
Secondary User	press()	press()	press()	search(type, name)
Guest User				search(type, name)

3.5 Boundary conditions



AdministratorUser can use ManageMedia which invokes AddMedia, DeleteMedia, ChangeNames. These cases are using for adding new medias to application, remove media from application or change the media names in the application.

AdministratorUser can use ManageUser which invokes CreateNewUser and DeleteUser. The CreateNewUser case adds a new secondary user to the system. The DeleteUser case removes any existing secondary users on the system.

AdministratorUser can use ChangeCosmetics. With this administrator user can change theme model and color, font size of application or specific tab, change the order of upper tab(last played, playlists, favorites), can set black and white mod about background, can set new equalizer settings, and some standardizing about small details.

INITIALIZING:

AdministratorUser initialize the system via create the first account. After the first initialization, other SecondaryUser can be added to system. And they can initialize the system on their own without the allow or audit of the AdministratorUser no more. Also GuestUser do not require for created by AdministratorUser and can initialize the system without allow of the Administrator.

In initializing stage, the LogIn and EnterUserInformation cases are used. At the LogIn case, first the user type is selected; then RegisteredUser entering the user information (EnterUserInformation) and initialize the system.

At the initializing the user interface represent the LogIn screen. After the initialization the user interface open the specific main screen for each user type.

TERMINATION:

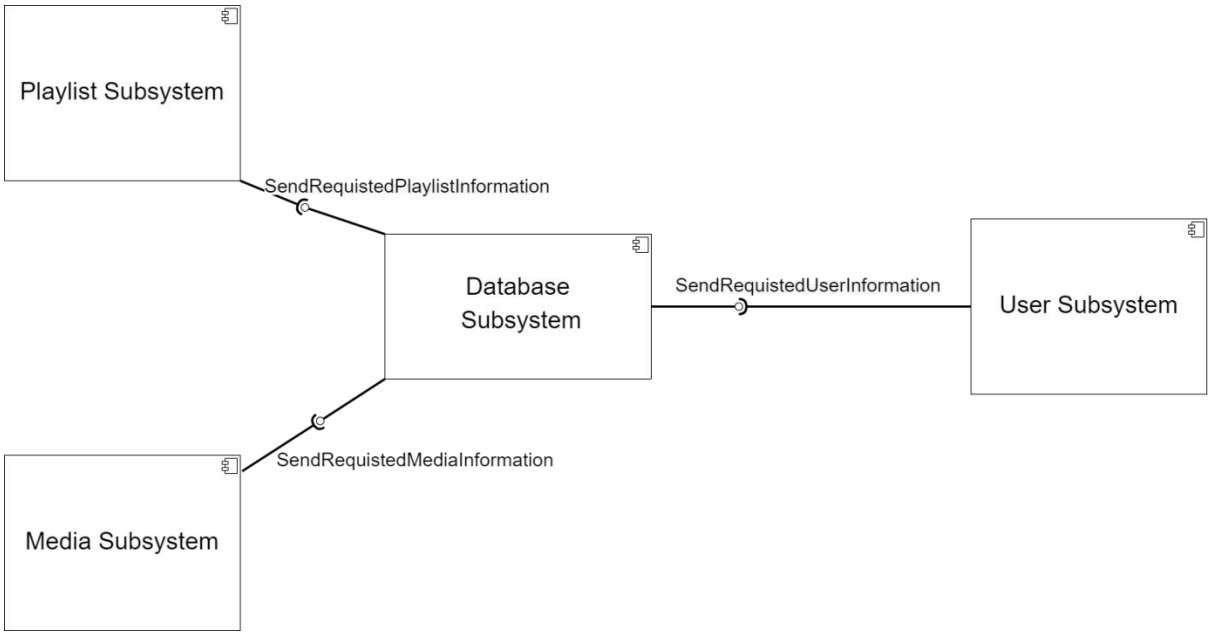
User Subsystem can terminate the application via LogOut case or exiting from the application by using the ExitButton which invoked by Button Subsystem. It is done by User just using the exiting from system via LogOut case or ExitButton. When the User don't quit from the application by not LogOut or use ExitButton, system don't stop the run along via StallingTime.

While quiting the program, after the press ExitButton; the system must ask to user a question like "Are you sure about quiting ?". In this way, the system preventing the user from losing data accidentally.

FAILURE:

When User accidentally exits from the application, the application don't lose the data for StallingTime. Initializing the application after the accidentally quit, the User can continue where User was at that moment.

4 Subsystem Services



Media Subsystem: Executes, displays and sorts media files. Configurates how the media file is executed.

Playlist Subsystem: Manages, adds, removes media files from playlists. This subsytem also creates and deletes playlists.

User Subsystem: There are 3 main account types, the authorization of these accounts is different from each other. The most authorized is administrator user. Second user has general authority. Guest user privileges are very limited. Also User subsystem manipulates application style; stores account and cosmetic info.

Button Subsystem: Manages user's interaction with the application. Acts as a bridge between user and the application.

Database Subsystem: Stores account, media and application cosmetics data. Manages communication about media files between other subsystems.

5 Glossary

Describe the definitions, acronyms, and abbreviations you used in your report.

ExitButton	Exit button closes the application when this button is pressed.
StallingTime	It is the situation when the system is actively running but does not receive any input from the User.
MediaFileStoreSubsystem:	All data that the administrator user addsto the system is stored in this subsystem. Other users can access the medias in the system with use this subsystem.
UserDBStoreSubsystem:	This subsystem is responsible for storing data belong to all user. The users communicates with the subsystemto do any operation in the application.

6 References

If you refer to other sources, list them here.

[1] https://github.com/BAU-the-Computerer/Project---OOAD/blob/master/doc/Group10_RAD-v3.pdf

[2] https://app.diagrams.net/#G1a5dMgkTiaHzU0cAH7K6KECURoasW-jUP

[3] *https://my.oschina.net/mavericsoung/blog/125884*

[4] *https://en.wikipedia.org/wiki/Media_Foundation*